Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2009 Proceedings

Americas Conference on Information Systems (AMCIS)

2009

A Comparison of Three Modes of Collaboration for Software Development

Mojgan Mohtashami
AID, Inc., mojgan@aidpe.com

Vassilka D. Kirova
Alcatel-Lucent, kirova@alcatel-lucent.com

Thomas J. Marlowe
Seton Hall University, marlowto@shu.edu

Fadi P. Deek
New Jersey Institute of Technology, fadi.deek@njit.edu

A Comparison of Three Modes of Collaboration for Software Development

Mojgan Mohtashami
AID. Inc.
mojgan@aidpe.com

Vassilka D. Kirova
Alcatel-Lucent
kirova@alcatel-lucent.com

Thomas J. Marlowe
Seton Hall University
marlowto@shu.edu

Fadi P. Deek
NJIT
Fadi.deek@njit.edu

ABSTRACT

In the current environment of global collaboration, multiple models for collaborative ventures have been introduced. We consider traditional, contractual interactions as well as three modes of collaboration—supply-chain management, a standard virtual organization, and inter-organizational collaborative software development (ICSD). After outlining multiple characteristics of these modes, and their strengths and weaknesses, we examine how to select an approach for a software project, and characterize non-software-development applications for which an ICSD approach may be appropriate. Each of these can then be used as input in selecting an optimal mode and practices for the project.

Keywords

Contractual ventures, inter-organizational collaboration, inter-organizational collaborative software development, management contingency processes, risk management, Supply Chain Management, virtual organization

INTRODUCTION

Collaboration is of increasing importance in the modern business environment. Research and development in many areas follows a similar pattern, with increasing connections between institutions in industry, government, and higher education.

Collaborative alliances provide numerous benefits. Alliances provide access to specialized knowledge and services that may not be needed on a continuing basis, or are more costly to develop and maintain than to access from others. They also tolerate fluctuations in resource demand, minimizing loss and cost, redistributing slack resources, limiting hiring and training delays resulting from either staff turnover, or the transient need for particular expertise. Finally, they support redistribution and reduction of risk, by broadening the market; providing experience with complementary and perhaps specialized managerial approaches; making refined risk analysis possible via multiple perspectives; and, as suggested above, accommodating schedule delays by recovering slack at collaborating partners.

In this paper, we contrast traditional contractual development with three modes of collaboration, primarily in the context of software development — Supply Chain Management [SCM], a selected model of Virtual Organizations [VO], and Inter-organizational Collaborative Software Development [ICSD], which blends aspects of standard software development with features and attributes of the other two collaborative modes.

Compared to the other modes, VO is a much broader category — ranging from a collaboration between a hotel chain and an airline to share promotions, through cooperation between component manufacturers to produce a home entertainment center, to an initiative involving multiple government, military, research and industrial organizations

in developing and producing a new space travel vehicle. The characteristics of virtual organizations are thus much harder to pin down precisely. To simplify discussion, we focus primarily but not exclusively on VO for development, although we note characteristics that differ across the range of virtual organizations.

In the rest of this paper, we consider the characteristics, advantages, and disadvantages of these modes. We then suggest appropriate modes for multi-organizational software projects, based on project characteristics, and characterize other collaborative ventures that can take advantage of the ICSD model. Finally, we review related work, give our conclusions, and sketch future investigations.

CONTRACTUAL DEVELOPMENT AND THREE MODES OF COLLABORATION

We first introduce some terminology. In any collaboration, the *project* is the total development effort from beginning to end; the *process* is the technical and management framework and methodology used by a project to create a product; and the *product* comprises the deliverables of a project. The *platform* is the environment for developing or deploying the product. A *component* is a part or module of the product, or possibly a tool used to develop it, managed by a collaborating *partner*.

Contractual Development [CD]

In contractual development, parts of a project or product are subcontracted, with tightly defined business and product requirements. Subcontractors need not be aware of the nature of the full project/product, and decision making is largely at the discretion of the principal. Business interactions are typically limited, and relate almost entirely to the contracted part, phase, or service, while technical interactions are generally restricted to improving the physical or logical interface between components and ensuring specification compliance.

Contractual development of software is strongly tied to traditional software engineering practices (Pressman 2005). A developer or integrator typically farms out specific services, tools, or data management tasks, most often to suppliers or vendors who have previously developed or provided similar services, or worked in the product domain. The contract necessarily relies on a nearly full specification of the component and its interface, requiring precisely specified behavior and performance, with minimal room for flexibility. From the principal's design perspective, the contracted component is effectively "plug-and-play".

Advantages include not only familiarity, but a clear legal and technical framework; disadvantages result from lack of flexibility, and lack of shared perspective, shared processes, collaborative risk analysis, and supplier autonomy.

Supply Chain Management [SCM]

SCM is the oldest and most established ongoing model for inter-organizational collaboration. Each collaborating organization is a value- or feature-adding agent in creation of a final product. Generalizing the assembly-line model, it provides a largely rigid and structured model for interaction: orders and payments flow upward, and products and invoices flow downward in an almost invariably acyclic supplier-consumer graph.

The performance and eventual success of the collaboration is highly correlated with partner performance, and both upstream and downstream nodes can be affected by problems at intermediate nodes. Success is also strongly affected by effectiveness of communication, coordination and collaboration.

Advantages include: a well-understood and predictable model with clear responsibilities, and long-lived if often intermittent relationships, allowing forecasting and optimization. Disadvantages largely relate to limits of the underlying model: vulnerability to single-point failures, poor response to evolution in market, infrastructure, or requirements, and limited resource flexibility.

Virtual Organizations [VO]

Several factors motivated the development of VO. Changing technology and increasing product complexity, result in larger projects, requiring more diversity in skill and knowledge, adaptability, stronger feedback, and timely interactions. SCM is not a perfect match for such development, given increasing globalization and customer sophistication, the need for customization, a greater need for services with non-linear interactions, both during design and in deployment, the wider market domain, and the need for specialized local knowledge and expertise.

VO is typically a better fit with the (hierarchical) team model. It is often short-lived, if repeatable, with partners being brought together for a highly specific reason.

Advantages, in addition to increased flexibility and adaptability, include a shared interest in the final product or project, and the possibility of unexpected contributions (via innovation or serendipity) to its development. Most importantly, perhaps, it allows substantial freedom in specifying components, even though once the high level system decomposition is made it remains largely frozen together with component interfaces (although components can be further divided hierarchically).

Inter-organizational Collaborative Software Development [ICSD]

Neither a short-lived VO nor SCM is a perfect match for software development, since software projects tend to be long-lived, requiring ongoing collaboration not only during design, but continuing on through maintenance and evolution; produce intellectual property rather than a physical "widget"; and have large numbers of diverse stakeholders. Moreover, the interaction between phases in software development is both more dynamic and less predictable.

Thus, ICSD has been developed as a specialization of VO that further inherits features from the other modes and provides capabilities for dealing with complexities that arise in software development. For example, it extends VO concepts with software development practices such as hierarchical change management, thus accommodating the longer lifetime and post-release product adaptation and evolution, as well as the typical imprecision and incompleteness of initial software requirements.

In addition, software components are typically highly interdependent and evolvable, and the development process is communication-intensive, requiring effective intercultural and inter-organizational communication, integrated and collaboration-aware management contingency policies and risk management (Barki 2001, Mohtashami-1 2006), and a clear process for mediation or arbitration.

At the technical end, ICSD is best based on modern object-oriented software engineering practices (Larman 2004), which provide natural support for flexibility and evolvable interfaces, hence for distributed development. However, to support autonomous collaborative development, further enhancements (e.g., multi-layered interfaces) are required to allow for component flexibility while assuring proper integration (Marlowe, Kirova 2008, Kirova, Marlowe 2008).

Advantages include the ability to use modern software engineering and software management approaches in development, supporting flexibility and evolution; disadvantages include a need for substantial and ongoing collaborative risk analysis, technical and management cohesion, streamlining, and communication, and reflection from all parties in the collaboration.

CHARACTERISTICS OF THE THREE MODES

In this section, the three modes are characterized for a number of major attributes. In response to the variability in VO characteristics, we consider a production-oriented VO; those of a service-oriented or marketing-oriented VO may be somewhat different.

Management Profile

In developing a strategic alliance, there are four steps to be considered: conceptualizing, pursuing, confirming, and implementing/continuing the alliance (Handfield, Nichols 1999). Even though these four phases pertain, to some extent, to all collaborative ventures, each phase assumes different and specific characteristics depending on the collaborative mode employed.

In SCM, management plays a vital role in initiating, establishing and defining the collaborative venture, as well as in negotiating the terms and rules of the collaboration. After establishment of the venture, the collaborative structure is largely predefined, fixed and encapsulated. Management intervention may still be required from time to time, to address problems and to tune the collaboration.

In a VO or ICSD venture, in contrast, the need for a management intervention and support remains strong throughout the project. Our analysis (Mohtashami 2006) corroborates the view that ongoing management support

and championship are factors that not only contribute substantially to success of inter-organizational communication, but also have a direct effect on the success of collaborative software development efforts. Management of VO and even more of ISCD must be concerned with: flexibility, adaptability of policies and environment, ability to capture needed expertise, need to extend the market domain, and need to share and combine resources. ICSD management will also need to be concerned with technical and risk management process issues, because of the nature of software development and required technical communication.

In SCM, the structure of management, distribution of power, and domains of authority among collaborating agencies are clearly defined. For VO and ICSD, on the other hand, there is an inherent tension between the interests of the partners and the interests of the collaboration, and a need for a process for negotiation and mediation/arbitration. However, as specified earlier, for collaborative ventures, the necessary management structure, management life cycle and distribution of responsibilities is not immediately evident, and is in fact an area ripe for extensive research.

Inter-Organizational Communication

SCM requires only limited, predictable, and highly structured communication, once the chain is in place. VO requires a high level of initial technical exchange to agree on component interfaces, and a high level of ongoing management exchange; the degree will of course vary with the type and complexity of the project, the degree of collaboration needed, and the collaborative experience of the partners. An ICSD project also requires high level of management and technical exchange, both initially and on an ongoing basis; the communication involves product, process and project aspects, and depends on extensive interpersonal exchange at all levels.

Process Coordination and Planning

SCM requires some (variable) effort to create the chain, but minimal, predictable and structured planning thereafter, except for the high cost and effort to replace a partner. VO typically requires more effort to create, and requires significant effort to maintain collaboration, coordination, and communication thereafter. An ICSD VO project requires in addition a higher level of technical collaboration and communication.

There are two other aspects of process coordination: risk management, discussed below, and schedule issues. Because of the largely linear structure of interactions in a typical supply chain, any lengthy interruption, bottleneck or delay will degrade performance of the entire chain, and an SCM venture must be concerned with streamlining across the chain. In a VO, there may be more leeway for transient problems. ICSD inherently supports redundancy and/or substitutability of work among partners as they regularly offer similar software expertise and development platforms. It also allows for much higher degree of parallelism in design/development task, although specialized and unduplicated domain or feature knowledge may be a constraint in some cases.

Product Coordination and Integration

All three modes require coordination. Coordination in SCM is largely temporal and limited to scheduling; in VO, product coordination is also needed to ensure compatibility. Finally, in ICSD, product and process coordination dominates, with the aim of assuring interface stability in the face of product evolution, and the satisfaction of end-to-end product and quality assurance requirements.

SCM integration is largely virtual—if there is a product, plug-and-play integration largely suffices, and organizational integration is not required. In a VO or ICSD, the integration is essential: the venture needs to operate as a single entity in seeking to fulfill the objectives of the collaboration. Since product integration is difficult even in a single-organization, single-site, single-team development of large software systems, ICSD must also incorporate significant product integration, with integration failures to be expected and leading to additional iterations of the development process.

Information Flow

SCM information flow is unidirectional for each kind of message—information, data and requests from buyer to supplier, and material, invoices and associated data from supplier to buyer. The type, frequency, and format of this information, once the supply chain has been established, are predictable and predefined, typically periodic with well-understood patterns and sequences. In a VO, bidirectional flow is needed to coordinate and integrate activities, and, as usual, in ICSD, a much heavier, bidirectional and iterative flow is needed to coordinate and integrate activities and assure product and process coordination.

Information Sharing

Basic SCM requires minimal information sharing—primarily contract-related, but an SCM-based collaboration may lead to a long-lasting strategic alliance, requiring partners to share confidential corporate information (for example, forecasts, point-of-sale data and inventory status), to deploy assets for joint projects, and to pursue process improvements.

The type and content of information that needs to flow across channels linking collaborating organizations in a VO or an ICSD alliance is, however, far wider and more critical. For a product-oriented VO and for ICSD partnerships in particular, information sharing requires more comprehensive, real-time sharing of information, including technical (both process and product), management, change, and risk information.

In ICSD, notably more process and product information needs to be shared due to typically incomplete initial requirements and the need for iterative refinement. Also the complexity of cross-organizational testing and validation, as well as maintenance and evolution require steady information exchange and coordination. Further, cross-organizational knowledge management aspects need to be addressed (Flynn 2008, Marlowe 2008).

Typical Lifetime

SCM ventures thrive on establishment of long-term partnerships, with continuous existence and planned activities. Switching partners is expensive, introducing turbulence and uncertainty in established policies and practices. In contrast, a VO's lifetime and activity are for the duration of the project, typically fairly short, ending with delivery or release, or failure, of the product (or the end of a specified term, for service-oriented and market-oriented VOs) (Cantu 1997). Joint venture alliances form, dissolve and reform continually, based on business needs.

ICSD lies somewhere in between. ICSD ventures require extended commitments, based on the complexity and long development time for large software projects, whose lifetime continues long after initial product release—studies have shown that the majority of project effort and lifespan occurs in post-release maintenance and evolution (Pressman 2005). As our analysis reveals, establishment of communication channels and protocols, and enhancement of trust, cultural familiarity and work familiarity, are critical to the success of ICSD. But such establishment demands tremendous resources and investment, and therefore encourages long term and stable relationship among collaborating partners.

Flexibility

Most successful SCM ventures need only minimal flexibility, but have low tolerance for problems and failures—compare inventory management system failures following natural disasters. A VO requires substantial flexibility in management practices and policies, but typically does not need elaborate mechanisms for negotiating and accommodating technical changes, once the initial component decomposition and interface descriptions are put in place. Finally, ICSD, because of the inherent incompleteness of initial requirements and initial risk analysis, needs high levels of management and technical flexibility and support for evolution.

Risk Management

SCM encapsulates partner activities, and benefits from having relatively well-understood risks at the interfaces, with the primary inter-organizational risks arising from shipping and communication infrastructure. Thus, risk management can largely be autonomous and local. Risk analysis relies heavily on historical data for (again, localized) forecasting. A global view will nonetheless still have benefits, primarily in estimating the impact of a common external problem, or of the overall effects of delays or bottlenecks at one partner.

In a VO, collaboration introduces new risks, due for example to miscommunication; the intensity of these new risks is very much domain-dependent. There is little if any historical data, and few past standardized or documented activities, to rely on—and even those may not be universally applicable. Rather, the novelty and variation in the nature of VO and ICSD ventures requires that risk management analyses and strategies must be defined and studied objectively and carefully. Finally, the sensitivity of the VO to the changing (business, social, economic and technical) environment implies an active, dynamic, and continuing risk analysis.

In ICSD, there are still higher risks arising from the more intense collaboration, and also the standard risks of software development, and risk management must be collaboration-aware and coordinated (Mohtashami 2006), and

highly aware of problems with cultural and work familiarity and quality of communication.

Management Intervention

In SCM, as usual, management intervention is needed primarily at the onset of collaboration, and otherwise only in case of serious problems. For a product-oriented VO, constant intervention may be required, depending in part on product domain. ICSD likewise requires constant intervention and support, and because of product flexibility and evolvability, together with intellectual property concerns, requires a project-long process for negotiation, and if necessary, mediation or arbitration.

Collaborative IT and Communication Support

IT is a key enabler of distributed projects, and even more important in cross-institutional collaboration. SCM requires standard, stable, and consistent support, of limited form and content, needs minimal support for group communication, and has more tolerance for minor IT delays. ICSD, on the other hand absolutely requires more extensive and flexible support not only for management coordination and planning, but for exchange of technical and process information. VO has needs intermediate between the other modes.

Organizational Learning Capacity

Not an issue for SCM, in which the universe is largely static, but a significant success factor for VO, especially for longer project lifetimes, complex domains, and/or ambitious goals; an absolute necessity for ICSD, with the need for adaptability and for process evolution.

Trust and Cultural Familiarity

This is not usually an issue for SCM, since exchanges are structured, formulaic, and largely culture-neutral. As noted, collaborations are highly structured and predictable, so once the supply chain is established, satisfactory delivery is sufficient to justify continued trust. Exchange of personal or informal information may improve executive investment, but is not necessary for successful continuation.

In a VO, in contrast, exchange of personal, informal, and cultural information aids in building trust between collaborators, and in providing a context for interpretation of partner communications and actions. Since VO requires organizational trust, organizations that prove trustworthy and successful with collaborations will be sought as partners. Moreover, success of collaboration also relies on effective communication, trust, and mutual understanding at technical and other process layers. Beyond VO, ICSD requires not only organizational, but also interpersonal trust, since the success of the product relies on detailed exchange and shared understanding of code, documents, and comments, and on often not easily assessed characteristics of the internals of other partners' components.

MODE SELECTION FOR SOFTWARE DEVELOPMENT PROJECTS

While ICSD offers a general approach for collaborative software development, it is a high-overhead, complex and intricate process. Many projects may not need the full ICSD structure, and one of the other approaches discussed may be less costly and equally or more appropriate.

For example, contractual development remains a reasonable choice when the project is clearly defined, the domain is well understood, decomposition into components is evident, and the components are minimally coupled, with clear and well-specified interfaces; it also helps if subcontractors are trusted, and cost and schedule are easy to determine.

The supply-chain model may be useful, at least by analogy, when partners are established experts, willing to enter an ongoing relationship; components (modules, tools, or services) are pre-existing, possibly requiring minor specializations, or easily fabricated from well-understood requirements, and integration fits an assembly-line or plug-and-play model with minimal expectation of cross-component iteration.

A virtual organization without all of the ICSD refinements and overhead may suffice when prospective partners are interested in producing a specific product or result, usually a minor variation of an existing system; the product can be decomposed into modules, tools and services, although possibly with more interaction than in the SCM case above; component characteristics are largely understood; partners are able to contribute specific expertise in

developing specific components, with minimal overlap of specializations; and there is minimal expectation of serious maintenance or evolution, or one partner is willing to take responsibility after release of the product.

A full-fledged ICSD collaboration may be indicated when faced with several of the following:

- a large complex software project, especially if component boundaries or the details of the interface cannot be frozen at project start;
- the project definition is imperfect and/or incomplete, and has substantial differences from past projects;
- the product requires instantiation of diverse expert domain or feature knowledge, especially if that knowledge should apply across components;
- there are a large number of heterogeneous stakeholders, not all of whom are partners to the collaboration;
- there are substantial risks related to trust, communication, culture, or innovation, especially if individual problems could impact multiple components, or coordinated risk management is otherwise desirable;
- there is a likelihood of multiple iterations, long-term maintenance and evolution, and/or the project is expected to have a very long lifetime;
- the project requires substantial innovation, in design and/or implementation/construction, or requires process innovation; or
- component development relies on substantial information from other components, process coordination has clear benefits, and significant management coordination of technical activities is required.

ICSD also depends on partner willingness to enter into this mode of collaboration and the feasibility of an initial interface specification sufficient to allow autonomous development.

ICSD AS AN APPROACH OUTSIDE OF SOFTWARE DEVELOPMENT

Many of the VO specializations of ICSD may be useful in other areas. The characteristics above, translated from the software universe to the product domain, are good indicators of this possibility. For example, consider distributed development of courses for a mathematics major program at a virtual university, such as New Jersey's Thomas Edison College (Edison 2009). For simplicity, assume that each course or two-course sequence is being individually designed, while the overall structure is being negotiated among participants.

The structure of the major is well-understood, although with alternatives; some topics may be covered in several different courses. Once course structure is largely determined, there are significant content/prerequisite issues that introduce dependencies in course design and implementation, requiring communication among their developers. The interface between Calculus II and Differential Equations may be flexible in coverage of special functions or of elementary differential equations in the former, and the need for previous coverage of infinite series in the latter; in addition, the latter course may rely on previously introduced software tools or approaches. But the interface should be able to assume some standard content on both ends, yet allow enhanced content on either end, with a process for arbitrating conflicts.

In addition, creating an appealing quality program entails investigating novel pedagogical techniques and combinations of media and delivery, including virtual classroom and delivery approaches, and also mathematical software and other tools for exploration. The major is expected to be long-lived—mathematics will be a possible major for decades. Finally, we can expect substantial adaptation and evolution, with changes in student interests, the delivery platform, the Internet and its access mechanisms, and supporting software, expected content and pedagogy in a mathematics degree, instructor specializations, and, more slowly, changes in course offerings and degree structure.

More generally, ICSD features apply to many virtual organizations: process-sensitive management; IT and communication infrastructure supporting management and technical exchange and coordination; flexible but constrained interfaces, collaborative risk management; incorporation of knowledge management in technical aspects; emphasis on trust, cultural familiarity, management support, and employee empowerment; collaborative

change management; and a process for negotiation and arbitration of differences and responsibilities for change.

RELATED WORK

The primary resources for the selection of the success factors and the modes of collaboration in ICSD, discussed here, are the pertinent literature and the results from a survey study of more than 80 CSD projects we have conducted. This study confirms the importance of effective information exchange and shows that (1) the effect of most risk factors on software quality is largely accounted for by the effect on communication, and (2) communication metrics are highly predictive of software quality (Mohtashami 2006).

In aforementioned study, three success factors: (1) Organizational Background, (2) Information Technology, and (3) Management Contingency Profile (MCP), were found to positively affect the success of ICSD projects. Of the three, MCP, including risk management, proved across a spectrum of analyses, to have the greatest influence on projects' success. ICSD requires new structures and management profile to both deal with the changed nature of the development process, and to address new and aggravated risk factors arising from the combination of collaboration and software development. The findings of our studies regarding the three success factors are largely consistent with the findings of literature on organizational behavior and SCM.

SCM is studied and used extensively, covering all aspects of single organization and some collaborative business management (SCM-Council 2002; Chopra, Meindl 2001). The notion of distributed collaboration demands close attention to socio-economic and cognitive issues, such as trust (Alexander 2002), cultural differences (Massey et al. 2001; Yoo, Ginzberg, Ahn 1999) , and knowledge sharing and management (Agresti 2000; Hustad 2004). VO in its various forms is discussed in (Schniederjans, Schniederjans, Schniederjans 2005). ICSD has been covered extensively in, for example, (Mohtashami 2006, Kirova, Marlowe 2008). For inter-organizational knowledge management, see (Jastroch 2008).

CONCLUSIONS AND FUTURE WORK

We have identified four modes of inter-organizational interaction used in software development—CD, SCM, VO, and ICSD—and have focused on exemplifying the three collaborative ones (SCM, VO and ICSD). We characterize software projects that can benefit from each approach, and conversely suggest how ICSD, currently limited to software development, can be brought to bear on classes of non-software-development projects.

Our current and future work on collaboration is channeled in three directions: extending the work on modes of collaboration presented in this paper, exploring knowledge management and information flow aspects at component and organizational interfaces, and continuing our examination of management policies for collaborative software development.

In particular, we plan to refine our approach for matching projects to collaboration modes and to evaluate the utility of structured combinations of modes, for example, using SCM where some components are developed and maintained using ICSD, and others are delivered collaboratively in a product-oriented VO. In support, we intend to extend our early work on metrics for successful collaboration to metrics that assesses readiness for collaboration and predisposition to effective collaboration in selected modes. We also plan to refine our work on collaborative and collaboration-aware risk management to reflect the specifics of different modes of collaboration discussed in this paper.

REFERENCES

1. Agresti, W. (2000) Knowledge Management, Advances in Computers, Vol. 53.
2. Alexander, P.M. (2002) Teamwork, Time, Trust and Information, Proc. of the 2002 Annual Research Conference of the South African Institute of Computer Science and Information Technologists on Enablement through technology, 65 – 74.
3. Barki, H., Rivard, S., Talbot, J. (Spring 2001) An Integrative Contingency Model of Software Project Risk Management, Journal of Management Information Systems, 17, 4, 37 – 70.
4. Chopra S., Meindl P. (2001) Supply Chain Management, Strategy, Planning and Operation; Prentice-Hall, Upper Saddle River, NJ.
5. Flynn, D., Brown E., Krieg R. (June 2008) A Method for Knowledge Management and Communication within and across Multidisciplinary Teams, Workshop on Knowledge Generation, Communication and Management (KGCM 2008).

6. Handfield, R. B., Nichols, E. L., Jr. (1999) Introduction to Supply Chain Management, Prentice-Hall, Upper Saddle River, NJ.
7. Hustad, E. (2004) Knowledge Networking in Global Organizations: The Transfer of Knowledge, Proc. of SIFMIS.
8. Jastroch, N. (June 2008) Adaptive Interenterprise Knowledge Management Systems, 2008 Workshop on Knowledge Generation, Communication and Management (KGCM 2008).
9. Kirova, V., Marlowe, T. (December 2008) Accommodating Dynamic Change in Collaborative Software Development (in French), **Génie Logiciel**, Numéro 87, pp. 15-25., Paris, France.
10. Larman, C. (2004).Applying UML and Design Patterns. 3rd ed., Prentice Hall, Pub.
11. Massey, A. P., Hung, Y. C. *et al* (2001) When culture and style aren't about clothes: perceptions of task-technology 'fit' in global virtual teams, Proc. of the 2001 International ACM SIGGROUP Conf. on Supporting Group Work, 207–213.
12. Marlowe, T., Kirova, V. (2008) Addressing Change in Collaborative Software Development: Process and Product Agility and Automated Traceability, Proc. of 12th World Multi-Conference on Systemics, Cybernetics and Informatics, 209-215.
13. Mohtashami-1, M., Marlowe, T., Kirova, V., Deek, F.P. (2006) Risk Management for Collaborative Software Development, Information Systems Management, 23, 4, 20-30.
14. Mohtashami, M. (2006) The Antecedents and Impacts of Information Processing Effectiveness in Inter-Organizational Collaborative Software Development, Ph.D. Thesis, Rutgers University.
15. Pressman, R. S. (2005) Software Engineering: A practitioner's approach. 6th ed., McGraw-Hill., Pub.
16. Schniederjans, M.J., Schniederjans, A. M., Schniederjans, S. G. (2006) Outsourcing and Insourcing in an International Context, M. E. Sharpe, Armonk, NY.
17. SCM-Council (July 2002). The SCOR ™ Model Collaboration Framework, A Proposal for including the aspects of external collaboration within the SCOR ™ Model. Whitepaper. Supply Chain Management Council.
18. Thomas Edison College, State of NJ, http://www.tesc.edu/. (Accessed 21 February 2009.)
19. Yoo, Y., Ginzberg, M. J.,. Ahn, J. H. (1999). A cross-cultural investigation of the use of knowledge management systems, Proc. of the 20th international conference on information systems, 501-505.